Recursion of g, λ $(i = 1, 2, \cdots)$

$$\gamma_i = -\rho_{i-1} - \tilde{g}_i \hat{r}_i$$

$$g_{i+1} = \begin{bmatrix} g_i + \dfrac{\gamma_i}{\lambda_i} \hat{g}_i^* \\ \dfrac{\gamma_i}{\lambda_i} \end{bmatrix}$$

$$\lambda_{i-1} = \lambda_i - \frac{|\gamma_i|^2}{\lambda_i}$$

The Solution

$$s_{n-1} = k \begin{bmatrix} 1 \\ g_n \end{bmatrix}, \qquad k \text{ arbitrary}$$

Solvability Constraint

$$\lambda_n = 0$$

Or, equivalently

$$1 + \tilde{\hat{r}}_n \hat{g}_n^* = 0$$

Reference

1. Zohar, S., "Toeplitz Matrix Inversion: The Algorithm of W. F. Trench," *J. Assn. Comp. Mach.*, Vol. 16, No. 4, pp. 592–601, October 1969.

7. Information Systems: Performance of Short Constraint Length Convolutional Codes and a Heuristic Code-Construction Algorithm, *J. W. Layland*

a. Introduction. The simulated bit-error performance of short constraint length convolutional codes discussed in SPS 37-54, Vol. III, pp. 171–177 has created considerable interest, both at JPL and elsewhere, in the application of Viterbi's optimal decoding algorithm to actual communications. However, the original simulation results included only three codes, thus providing only a part of the information needed to make a decision on the design constraint length of a hardware optimum convolutional decoder. The decoder would be able to decode at least one code of each constraint length less than or equal to the design constraint length, but none larger.

This article presents simulated bit-error probability curves for convolutional codes of constraint lengths K = 3 to K = 10. The K = 3 and K = 4 codes are those discussed previously (SPS 37-54, Vol. III, pp. 171–177 and SPS 37-58, Vol. III, pp. 50–55). All codes of longer constraint length were constructed using a hill-climbing algorithm discussed in *Paragraph b* of this article. The simulated bit-error rates were obtained using the software optimum convolutional decoder described in SPS 37-62, Vol. II, pp. 61–66.

b. The code-construction algorithm. A convolutional code with constraint length K and rate 1/V is normally represented as a shift register of length K coupled with V multi-input mod-2 adders. Each information bit to be encoded is shifted into the shift register and the outputs of the V adders are sampled and transmitted sequentially. Such a code can be made into a block code of h information bits and $V(K + h - 1)$ channel symbols by requiring that all information bits before time 1 and after time h be identically zero. This is the technique used to determine the best known bounds to the free-distance of a convolutional code. (SPS 37-50, Vol. III, pp. 248–252.)

In a similar manner, one may approximate the bit-error probability of the convolutional code by considering only those error patterns that differ from the true sequence only in h or less consecutive positions. If the contribution to the bit-error probability from error events of length longer than h is sufficiently small, this will be a good approximation to the bit error probability. Since the free-distance bound becomes increasingly loose as h increases beyond a critical value, there seems to be some heuristic justification in assuming that there is a value of h, not too large, such that the contribution to the bit-error probability from error events longer than h is negligible.

Let I_i represent a bit sequence that is the binary expansion of the integer i, $C * I_i$ represent the coder output sequence corresponding to the input I_i, and $W_H(x)$ be the Hamming weight of the sequence x. Then a "union bound" to this approximate bit-error probability is given by (Ref. 1)

$$P_e^h \leq \sum_{\substack{i=1 \\ i \text{ odd}}}^{2^h - 1} W_H(I_i) \operatorname{erfc}\left(\frac{W_H(C * I_i) E_b}{V \cdot N_0}\right) \qquad (1)$$

where E_b is bit energy, and N_0 is the one-sided noise spectral density, assumed white and Gaussian. This bound, in turn, is upper bounded by

$$P_e^h \leq P_e'^h = \sum_{\substack{i=1 \\ i \text{ odd}}}^{2^h-1} W_H(I_i) \exp\left(\frac{-W_H(C*I_i)E_b}{V \cdot N_0}\right) \qquad (2)$$

The code construction technique discussed here uses $P_e'^h$ as a measure of the performance of the code. A hill-climbing algorithm is given that makes changes in the code's parity check equations; these changes effect the steepest descent in $P_e'^h$. The specific algorithm is as follows: Select a set of parity check equations from which to begin. This starting point may be arbitrary, but the most success has been achieved using check equations that are either all "zeros," or else have "ones" at the first and Kth positions only. $P_e'^h$ is computed for this starting code and for all $V \cdot K$ alternate codes within a radius-one Hamming neighborhood of the original code; i.e., all codes that differ from the original code in exactly one position. The code with the smallest $P_e'^h$ is selected as the starting code for the next iteration. The process terminates when none of the alternate codes have smaller $P_e'^h$ than does the base code of that iteration. No provision is made to prevent catastrophic error propagation in the constructed code. If the final code is catastrophic, the best alternate code from the last iteration is usually the desired code. If not, a small amount of trial-and-error searching will locate a good non-catastrophic code within a radius-two Hamming neighborhood of the terminal code.

Construction of the codes discussed in this article has been performed in all cases with the block-size $h = K$, $K+1$, or $K-2$, and with $\exp\{-E_b\ V \cdot N_0\} = 0.1$. When starting from check equations with very few "ones," the algorithm has been observed to always add "ones," resulting in termination in approximately $KV/2$ steps, after examining about $\frac{1}{2}(KV)^2$ codes, a small subset of the 2^{KV} possibilities. Since the algorithm uses $P_e'^h$ only for selecting a code change, $P_e'^h$ need be completely computed for only one code, with $P_e'^j, j \leq h$, computed for all others. The index j need only be large enough to guarantee that these codes are poorer than the chosen code.

The results obtained with this algorithm are aptly illustrated by Fig. 26, which compares the simulated bit-error probability of the $K = 8$, $V = 3$ code constructed with the algorithm and the $K = 8$, $V = 3$ code constructed by J. Odenwalder using a global search of all codes with a free-distance criterion as code performance (Ref. 1). The algorithm is thus not only an easy way to construct reasonably good codes, but perhaps the best way to construct codes for the bit-error probabilities presently of interest in space communications.

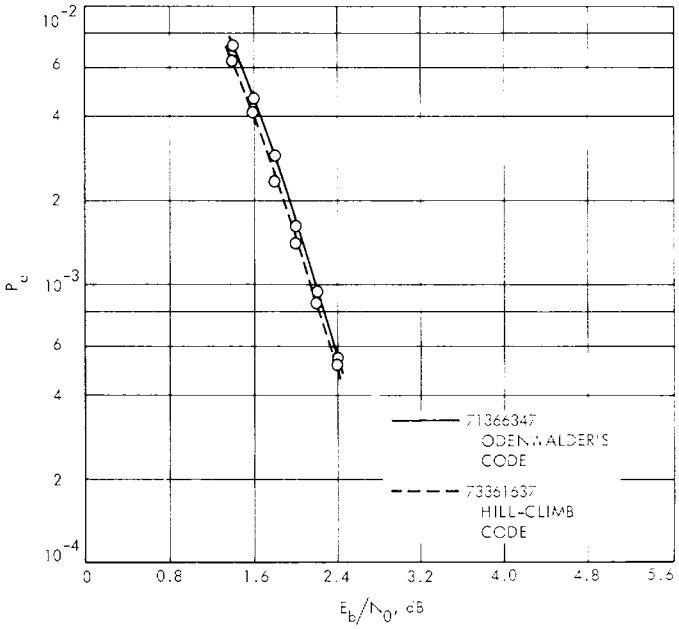

Fig. 26. Bit error probabilities of two $K = 8$, $V = 3$ codes

c. Simulation results. Figure 27 shows the simulated bit-error rate for eight convolutional codes with $K = 3$ to $K = 10$, when decoded by Viterbi's optimal decoding algorithm. The random noise was produced by the multiplicative-congruential generator ultimately adopted by Heller (SPS 37-56, Vol. III, p. 83). Codes in this figure are represented in octal, e.g., '313' represents the $K = 3$, $V = 2$ code which has

$$\begin{pmatrix} 101 \\ 111 \end{pmatrix}$$

for its parity-check equations, and '7567' represents the $K = 4$, $V = 3$ code,

$$\begin{pmatrix} 1111 \\ 1011 \\ 1101 \end{pmatrix}$$

As expected, bit-error performance is monotonic improving with constraint length, but the rate of improvement diminishes rapidly at $K = 8$ or $K = 10$.

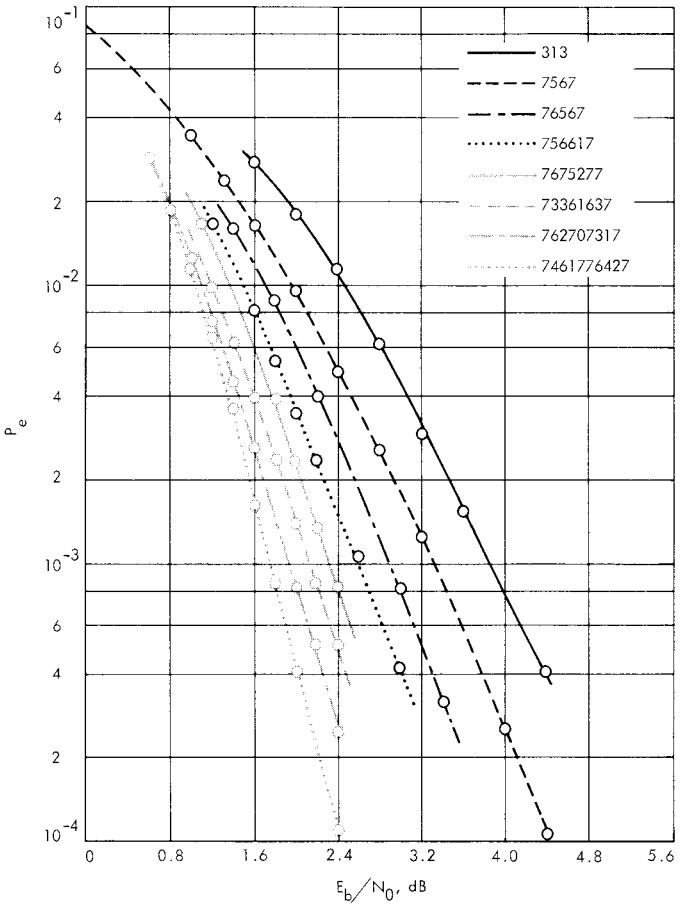

Fig. 27. Bit error probabilities of eight convolutional codes, $K = 3$ to $K = 10$

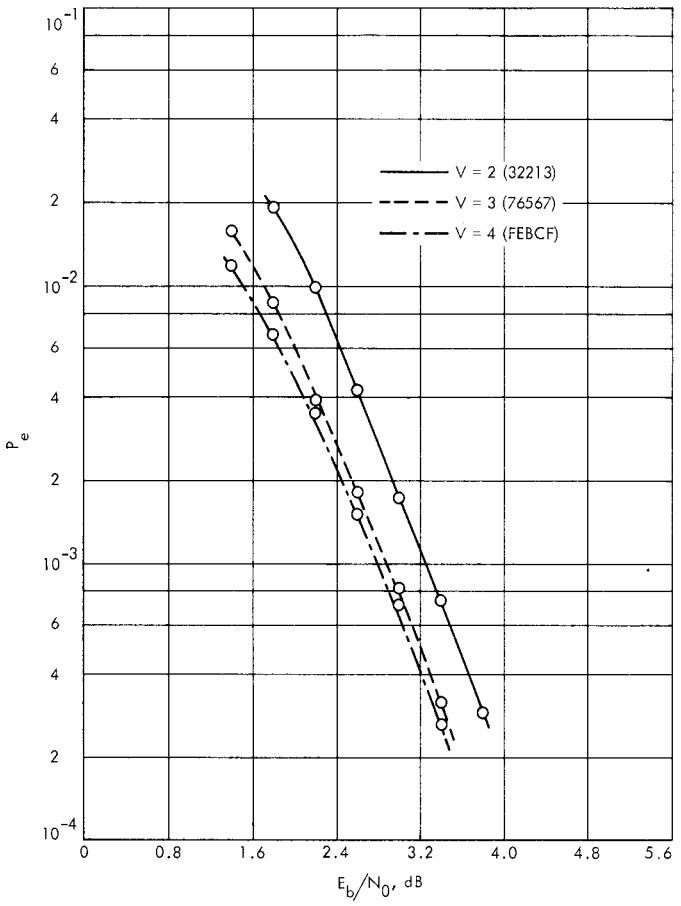

Fig. 28. Bit error probabilities of $K = 5$ codes of rate $V = 2$, $V = 3$, and $V = 4$

The effect of the code rate is illustrated in Fig. 28, which shows simulated bit-error rates for three $K = 5$ codes with rates $V = 2$, $V = 3$, and $V = 4$. The rate $V = 4$ code is given in hexadecimal notation with the symbols 0 to 9, A to F representing integers 0 to 15. The variation of the bit-error rate with code rate occurs similarly at other constraint lengths and agrees well with the behavior of the noisy-channel error rate exponent (Ref. 2).

d. Decoder complexity. An interesting view of the value of coding may be obtained by examining the "complexity" needed to achieve a design-goal bit-error rate as a function of the E_b/N_0 (Refs. 3 and 4). The complexity of an optimum convolutional decoder is approximately

$$\chi = 5K \cdot 2^{K-1} + \left(2 + \lfloor \log_2\left(\frac{KV}{2}\right) \rfloor + Q\right) 2^K + (\lfloor \log_2 (V) \rfloor + Q)\, 2^{V-1} \quad (3)$$

The notation $\lfloor x \rfloor$ represents the least integer greater than x and Q is the number of bits of quantization per symbol. Only the major components of the decoder are contained in Eq. (3). The first term is survivor storage, the second represents the state metric, and the last is caused by computing branch correlations.

Figure 29 shows the complexity as a function of the E_b/N_0 needed to achieve a 5×10^{-3} bit-error rate for the eight codes of Fig. 27. When presented in this form, the codes with $3 < K < 6$ appear to be the most desirable. For $K > 7$, the decoder complexity increases at a much faster rate than the required E_b/N_0 decreases.

Also shown in Fig. 29 for comparison is the complexity of the decoder for a bi-orthogonal block code at 5×10^{-3} bit-error probability. The block decoder is the optimally organized "Green machine" (SPS 37-39, Vol. IV, pp. 247–252). For both the block and convolutional decoders, 4-bit channel symbol quantization is assumed. At any fixed but

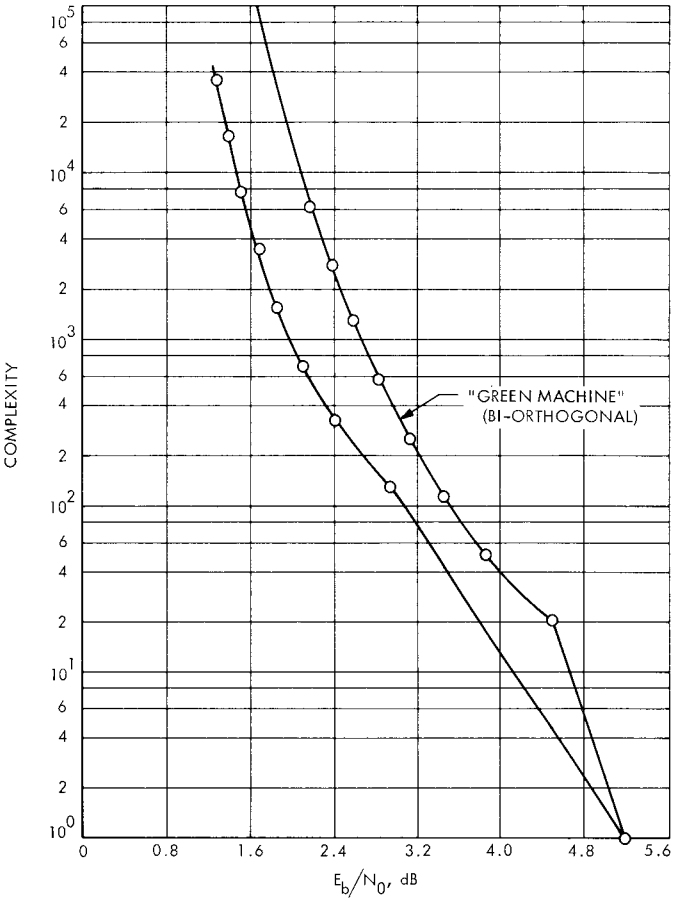

Fig. 29. Variation of decoder complexity with E_b/N_0 needed to achieve a 5×10^{-3} bit-error probability for the eight codes of Fig. 27

moderately large complexity, the convolutional codes require about 0.6 dB less E_b/N_0 to achieve the 5×10^{-3} bit-error probability than do the comparable block codes.

References

1. Odenwalder, J. P., *Optimal Decoding of Convolutional Codes*, Ph.D. dissertation. University of California at Los Angeles, Los Angeles, Calif., 1970.

2. Jacobs, I. M., "Sequential Decoding for Efficient Communications from Deep Space," *IEEE Trans. Commun. Technol.*, Vol. COM-15, pp. 492–501, Aug. 1967.

3. Savage, J. E., "Complexity of Decoders: Part I—Classes of Decoding Rules," *IEEE Trans. Inform. Theory*, Vol. IT-15, No. 6, Nov. 1969.

4. Savage, J. E., "Complexity of Decoders: Part II—Computational Complexity," Brown University Report, Providence, R.I., July 1969.

8. Information Systems: Synchronizability of Convolutional Codes, *J. W. Layland*

a. Introduction. There has been considerable interest recently in the practical application of Viterbi's optimal algorithm for decoding convolutional codes. Much of this interest arose as a result of the simulation work performed by Heller (SPS 37-54, Vol. III, pp. 171–177). These simulations, and most others, have assumed that branch synchronization is known to the decoder; i.e., that the decoder is informed as to which subgenerator of the code generates each symbol, and need not determine this from the code itself. Final coder synchronization is then achieved automatically by the decoding algorithm itself.

Although branch synchronization could be established by the use of a sync sequence periodically inserted into the data, it is much more desirable to use some property of the code itself for this purpose. For a block code, the comma-free property provides a sufficient means to establish synchronization in the *Mariner* Mars 1969 High-Rate Telemetry System. It seems intuitive that the subgenerators of any good code will be dissimilar enough that improper branch synchronization would be easily identified statistically, using the assumption that the data source is random. The truth of this notion is demonstrated by the simulation results presented in *Paragraph e* of this article. Synchronization can be achieved with more certainty, however, if some property can be found which will guarantee that only a finite number of code symbols need to be examined to identify a branch sync error. *Paragraphs b* and *c* determine the conditions that must be placed upon a convolutional code for this property to exist.

b. Convolutional codes. In its *standard form*, a constraint length K convolutional code with rate $1/V$ is represented as a shift register of length K, coupled with V parity-check adders. Each data bit to be encoded is shifted into the coder register and the oldest bit therein is shifted out. The V parity-check adders are then sampled sequentially and these binary symbols are transmitted through a channel. Figure 30 shows the encoder, channel, and decoder and the coder state diagram for a constraint length 3, rate ½ code. The output symbols for each state transition appear above each branch, and the input data bit appears below.

The channel adds a sample of white gaussian noise to each symbol. Since the channel does not explicitly provide synchronization information, the first symbol received by